LANGUAGE COMPILERS: SOME PRACTICAL OBSERVATIONS - Part 2: TURBO PASCAL

by A.C.L. Zelmer, Director, International Communications Institute
Box 8268, Station F, EDMONTON, AB T6H 4P1, CANADA

This series of three articles provides an introductory look at three different language compilers for the CP/M operating system. The first article looked at BASIC/Z, the BASIC compiler supported by the BASIC/Z CORNER. This article continues with an examination of the same program written using the Turbo Pascal compiler.

The program chosen for this discussion changes my Epson MX-100 printer from normal printing mode to condensed and back. The program uses many of the most useful aspects of any programming language, and the completed programs should be compared to see some of the differences among implementations.

The Pascal Language

Pascal was designed as a teaching language. It has elements common to many languages, uses a fairly formal control structure, and is currently the language of choice in many institutions for teaching as a first programming language.

As you will note in the program which follows, Pascal requires that you list (declare) all of your variables before using them. This and the lack of line-numbers will often confuse a BASIC programmer changing over to Pascal. Subroutines exist as procedures that are called when required as in the following example:

```
Switch(leftvar,rightvar);
```

Pascal allows the user to write multi-line functions that are called in a very similar manner to BASIC function calls. In addition, Pascal allows recursion, or the ability for a procedure or function to call itself.

A Pascal program consists mainly of a series of function and procedure calls. The procedures exist to solve specific problem steps and can call other procedures, functions or themselves as required. Since each procedure is a separate entity, Pascal programs should be quite easy to modify, and complicated programs can be built using previously tested routines (procedures) combined in a new program. The GOTO statement is seldom used, being replaced by a number of other more useful control statements.

I was introduced to Pascal with JRT's seemingly impossible offer of a few years back. At that time it seemed to be almost to good to be true that you could get a language compiler for less than $50. The local college here in Edmonton still uses the JRT Pascal compiler for teaching one of their classes, but students are advised that the compiler regularly 'goes out to lunch', and development work is slow.

I was therefore quite sceptical when Borland introduced Turbo Pascal, a compiler and editing system for the same kind of price as JRT's earlier failure. I was, however, quite desperate to write several demonstration programs for an introductory computer literacy class. These programs needed to use the computer, be simple, not games, and be bombproof (i.e.: the students shouldn't be able to change the programs). Assembly language programs would have been possible, however I didn't want to learn 8088 assembly language codes (the classes used IBM-PCs). The Pascal compiler seemed ideal and I took the chance (The cost was a major factor, I didn't want to use my own money to buy an expensive compiler for my employer's computer!).

TURBO Pascal

Borland offers TURBO Pascal for both CP/M-80 and 16 bit machines. On 16 bit machines the implementation includes colours, graphics, windows, sound and an option to use the 8087 math chip. CP/M users must be satisfied with a very adequate implementation that follows Standard Pascal with only a few differences. The system includes an editor (programs can also be written using WordStar or similar editors in non-document mode) and a compiler cum program executor. Programs are normally written and tested in memory, then a final compilation is made to produce a .COM file.

The TURBO system is very well integrated. When the compiler discovers an error in program syntax, the editor is invoked and the user is put into the program at the point of the error. Compilation is very fast. Even when I have made a number of errors, I can usually bounce in and out of the compile/edit/compile routines as fast as I can hit the keys (I am a two-finger typist, it takes me longer to type than to think).

I have only a couple of real problems with the TURBO package. First, the demonstration spreadsheet that comes with the compiler contains an error in the rounding routines so that columns don't always add up. Borland has thoughtgully provided the source code for this demo to assist programmers in understanding the use of Pascal. I would have thought, however, that a bug this basic in a demonstration program would have been fixed. Second, the source code print utility has a bug that miscounts the number of lines on a page of output. Again, something that should have been fixed before release.

Please remember that this isn't meant to be a review of this product. I haven't used other Pascal compilers to be able to make a comparison (I never did have enough time to get the JRT compiler to work). There have been a number of reviews of this and the other Borland products in the micro press. Consult them for details.

The Program

The combination of RUN/Z and the compiled BASIC/Z version of this program required 26K of disk space (see Part 1). The TURBO compiler's .COM file took 10K. It should be noted that this is about the minimum size of a .COM file generated by the TURBO compiler. The BASIC/Z system used a run-time module of 24K plus a 2K program-specific file. Thus ten small BASIC/Z programs might

take only 44K ((10*2)+24), whereas the same 10 programs written with this Pascal compiler would take 100K (10*10). The advantage of the Pascal .COM files is, of course, that they will run alone without any other required files.

As well, the TURBO compiler can be used for very large programs as it will allow the use of overlay files. [BASIC programmers usually get around the lack of program overlays by CHAINing from one program segment to another. The overlay system simplifies some aspects of large program development by moving partial program segments in and out (overlays) of the main program code as required. Variable use and program work space use is simpler with overlaid program segments, and program operation should be quicker.]

In the program below, the most noteworthy aspects of the code are probably the use of the 'Case' and 'do While' statements. The Case statement replaces a series of IF statements and includes an 'else' option if no match occurs. The main program control is based on the value of the variable 'Response'. The variable is checked as control enters the loop, all following statements are executed if the match fails, and the loop ends (hopefully logically) with an 'end' statement. The loop is bypassed if the match succeeds and execution continues with the next line after the end statement.

In this program the collection of end statements on the last three lines represents first the end of the Case statement, next the end of the While segment, and finally the end of the overall program. Note that indentation of program lines visually shows the level of control for any statement. As well, the program itself and all segments start with a 'begin' statement and finish with an 'end' statement. Finally, the Read() function corresponds to the INPUT statement in BASIC, and the colon/equal combination (:=) is the Pascal assignment operator.

This program was too short to illustrate any user-defined procedure calls, however the Read() function illustrates their use. Upcase() is a function call.

Pascal was designed to teach 'structured programming'. Structured programming is probably the exact opposite of a badly written 'spagetti bowl' of BASIC code. Certainly in my experience, Pascal is usable for general purpose programming and encourages me to be much more careful in my program design.

```
program PRINTER;
                                { Change the MX-100 print mode }
                                { A.C.L. ZELMER, Feb 85 }
                                { for Turbo Pascal Compiler }

Var
   Response:        Char;

begin
  While Response <> 'X' do
  begin
   ClrScr;
   Writeln('PRINTER'); Writeln;
```

```
  Write('This program shifts the MX-100 printer.  ');
  Writeln('The printer must be ON;');
  Writeln;
  Writeln('      for normal type              enter N');
  Writeln('      for condensed type           enter C');
  Writeln('      to eXit (quit)              enter X');
  Write('and press the <ENTER> or <RETURN> key.  :');

  Read(Response);
     Response := UpCase(Response);

  Case Response of
     'N' : begin
             Write(Lst,^R);
           end;
     'C' : begin
             Write(Lst,^O);
           end;
     'X' : ;
   else
     Writeln;
     Writeln('Please enter one of the specified responses.');
     Writeln('Press the <ENTER> or <RETURN> key to continue.');
     Read(Response);
       Response := UpCase(Response);  {x entered will exit}
  end;                                {Case Response}
 end;                                 {do loop}
end.                                  {Program PRINTER}
```

This was the first program that I tried writing in Pascal. It took me roughly a day of work to test out the routines and complete the program. Of course, this is in addition to the time that it took for me to read the manual, etc. The TURBO Pascal manual is not a tutorial for the language, I have found that the book 'Oh! Pascal!', by Doug Cooper and Michael Clancy (New York: W. W. Norton, 1982), is quite useful as a self-learning tool. The book 'Introduction to Pascal', Second Edition, by Neill Graham (St. Paul: West Publishing, 1983), has provided a number of useful routines for sorting, searching, file handling, etc.

Next time I'll look at the C/80 compiler from The Software Toolworks.

.....

MDOS UTILITIES ON MUG LIBRARY DISK 907

Burks Smith has decided to release his MDOS and Micropolis Basic utilities SMASH, BAS>LIN, LIN>BAS, and VARLIST to the public domain.

For maximum utility to the user, 8080 source code has been provided to permit user modification of the programs for special uses, if desired. Study of this code will give you valuable insight on the MDOS system and the use of the

built-in subroutines. The programs are also provided in executable object files, so you can run them without worrying about editing and assembling source files.

INSTRUCTIONS FOR USE

The following paragraphs describe the operation and syntax of each of Burks' programs on MUG library disk 907. Following the established Micropolis conventions, information in <> symbols indicates parameters that are to be provided by the operator. Information in [] symbols indicates optional parameters which, if entered, override "default" values assumed by the program. In both cases, the <> or [] symbols are not part of the command and should not be entered. Information not inside these symbols must be entered. A default drive of 0 is assumed if no unit number is entered.

SMASH BASIC PROGRAMS

The SMASH routine removes remarks and non-significant spaces from the Micropolis Basic program referenced by <source file> and creates a "smashed" version as the file <destination file>. This utility is useful for reducing the memory requirements of a Micropolis Basic program that contains REM or ! statements and spaces that make the program easier to read. The resulting program is difficult to read, but is of the smallest size that will still execute. No line numbers are removed, so remarks may still be used as entry points in the program.

Command syntax is:

SMASH "[unit:]<source file>" "[unit:]<destination file>"

Limitations: Since the resulting code is difficult to read, use this only for debugged programs., and save the fully commented form for reference. In some rare cases, removing spaces may produce ambiguous statements that may cause syntax errors in the resulting program. For example, the statement: PUT 2 R, S, T when "smashed" becomes PUT2R,S,T. Basic would interpret the 2R in the "smashed" version as a lead-in to a file number expressed in Radix 2 (binary) format, but when the first comma is encountered it assumes it has found a syntax error. The meaning can be made clear simply by inserting a single space after the file number. This is the only type of error that has been encountered as a result of using the program.

BASIC TO LINEEDIT CONVERSION

The BASIC TO LINEEDIT routine converts the Micropolis Basic program file referenced by <Basic file> to a LINEEDIT compatible text file of the name <text file>. The program expands one-byte Basic command and function tokens to ASCII text and generates LINEEDIT line numbers with proper compatibility. This program is useful when a text copy of a Basic program is needed for transmission via telecommunications or when global search and change features of LINEEDIT are desired.

Command syntax is:

BAS>LIN "[unit:]<Basic file>" "[unit:]<text file>"

Limitations: Basic programs allow a line length of up to 250 characters, while LINEEDIT only allows a maximum of 132 characters. Therefore, it is impossible to convert a basic line exceeding 132 characters to LINEEDIT format. If any line in the Basic program exceeds 132 characters, the program will abort with a PARM ERR message. It is recommended that long lines in the Basic program be broken into smaller lines, or deleted before performing the conversion. Note that LINEEDIT treats everything as text and line references in a Basic program have no meaning to it.

LINEEDIT TO BASIC CONVERSION

The LINEEDIT TO BASIC routine converts a LINEEDIT text file referenced by <text file> to a Micropolis Basic program file referenced by <Basic file>. The program compresses Basic commands and functions in ASCII to one-byte tokens and strips LINEEDIT line numbers. The program is useful for recovering files that had been previously converted to text by the BAS>LIN program and has many useful applications in converting programs from other languages to Micropolis Basic.

Command syntax is:

LIN>BAS "[unit:]<text file>" "[unit:]<Basic file>"

Limitations: This is a "dumb" conversion program and it assumes that the original text file is of proper Basic syntax and all line numbers are correct and in ascending order. It will try to convert any file produced by LINEEDIT, if asked, but the results may contain garbage that Basic can not recognize. Make sure that your source file is correct before using this program.

BASIC VARIABLE LISTER

The BASIC VARIABLE LISTER lists, on the console, all the variables and dimensioned arrays used in the Micropolis Basic program referenced by <filename>. If the optional parameter [1] is included, the program leaves a blank line in the listing for letters that have not been assigned as variables.

Command syntax is:

VARLIST "[unit:]<filename>" [1]

Limitations: The program includes the user-defined function names from DEF FNx or DEF FAx commands as variable names. It will report dimensioned arrays with calculated dimensions, but the dimension itself may appear as garbage. If printer output is desired, console data must be sent to the list device with the ASSIGN command.

.....

MDOS LIBRARY UPDATE

MUG MDOS Library Disk 907, Revision 0, May 85
MISCELLANEOUS

NAME	TYP	RV	SZE	CAT	DATE	AUTHOR/DESCRIPTION
S/SMASH	SRC	00	013	UTL	0585	Smith, B. Compresses Basic programs.
SMASH	SYS	00	003	UTL	0585	Smith, B. Executable version of above.
S/BAS>LIN	SRC	00	026	UTL	0585	Smith, B. Converts Basic to LINEEDIT form.
BAS>LIN	SYS	00	005	UTL	0585	Smith, B. Execuatable version of above.
S/LIN>BAS	SRC	00	024	UTL	0585	Smith, B. Converts LINEEDIT to Basic form.
LIN>BAS	SYS	00	005	UTL	0585	Smith, B. Executable version of above.
S/VARLIST	SRC	00	025	UTL	0585	Smith, B. Lists all Basic program variables.
VARLIST	SYS	00	004	UTL	0585	Smith, B. Executable version of above.
SYSQ1	SRC	00	005		0585	Smith, B. System references used for above source files.
SYSQ2	SRC	00	007		0585	Smith, B.
CALC-2	BAS	00	00F		0585	Hershell, G. A Spreadsheet (SuperCalc/Multi-Plan) program.
NAVIGATE	BAS	00	008		0585	Hershell, G. Calculates Distance (in nautical and statute miles, and kilometers) and compass bearing between two points which have been specified in degrees, minutes and seconds of longitude and latitude.
BOOT	SRC	00	01F		0585	James, B. Source for Micropolis bootstrap.
MOLOS	SRC	00	026		0585	James, B. MDOS version of SOL's SOLOS ROM.
VIDIO	SYS	00	004		0585	James, B.
PROM-SRC	SRC	00	061		0585	James, B. For SOL.
BURNPROM	SYS	00	00F		0585	James, B.
SOLOS	SRC	00	057		0585	James, B. Source for SOL's ROM.
BASE	BAS	00	005		0585	James, B. Converts numbers between bases.
DAY	BAS	00	005		0585	James, B. Computes a day of week from calandar date.
FINANCE	BAS	00	013		0585	James, B. Calculates loan & investment values.
CHESS	SYS	00	020		0585	James, B. A chess game for the SOL.
SUPERPEEK	BAS	00	008		0585	James, B. This, and the following three files, are routines for decoding OP-codes from memory.
PEEK	BAS	00	00B		0585	James, B.
PEEK1	BAS	00	00B		0585	James, B.
OPCODE	DAT	00	100		0585	James, B.
TARGET	SYS	00	011		0585	James, B. A great video game for the SOL.
AMO-VIEW	BAS	00	004		0585	James, B. Calculates loan values.
GRAPH	BAS	00	00C		0585	James, B. Creates bar graphs.
MENU	BAS	00	014		0585	Phillips, J. Database Manager.
TEST	DAT	00	005		0585	Phillips, J. Test file for above.

.....

LETTERS

SOL LOOK-ALIKES

An outfit in Kenosha, Wisconsin called Manu-tronics (9115 26th Avenue, ZIP 53140 - Phone (414) 694-7700) got stuck in a bad deal. As a result of that deal, they constructed a couple hundred computers like the SOL, except with 64K RAM and a few other goodies, designed by the guy who designed the SOL. This computer is (was) called "The Expander." They want to get rid of the stock at cost. I paid them $199 for one, and happened to get the last one set up for use in this country. The remainder are set up for use in Sweden--that is, they have a few peculiar characters on the keyboard chip. Because of that, they will sell for $75 apiece. These computers are built to use with Micropolis drives, but Manu-tronics has no drive boards and also no operating systems (and that means no Micropolis operating system) at all. They got me going with CP/M so that I could (eventually) transfer my ASCII files to my new Zenith Z-100. Now, if someone could get a Micropolis system going, Expanders should be fine for back-up, at least. Anyone interested should contact Leonard Garofalo.

Jack Peterson, 660 Forest Grove Circle, Brookfield WI 53005

.....

VECTOR SERVICE

Chapman Computer, Inc., has for some time offered a Vector Graphic board refurbishment program. We would like to extend that offering to your User's Group. Since 1982, we have been supporting a large number of Vector Graphic computers and related peripherals in the Georgia/South Eastern area. We will furnish a repair/refurbishment price list upon request. This list is typically less expensive than normal Vector Graphic end user prices. References are also available.

All parts are warrantied for a minimum of thirty days. An expedite service, for a nominal fee, is also available.

If you are interested, have any questions, or if we may be of help, please call.

Hollye M. Golash, Chapman Computers, Inc., 1901 Powers Ferry Road, Suite 210, Marietta, Georgia 30067 - Phone (404) 951-1913.

.....

TANDON DRIVES FOR THE 2600

I don't know much about hardware, including my TANDON 104M 77-Track 100 TPI disk drives which you advertised in the December 84 MUG Letter, but I think it would be prudent for me to have a backup drive available just in case one of mine goes down some day when I really need it. (I did replace a drive motor in

one of my drives the other day, following telephoned instructions from my former dealer in Syracuse -- so I could at least accomplish that.)

But I'm not sure I sufficiently understand what you mean by a "replacement drive" with "no power supply" or cabinet. The power comes from within the computer terminal body, does it not ? -- so one needs only to hook up, in the drive cabinet, the small plug connection to that source?

And it would not do to buy just a "single-drive Sub-system" (or a double-drive Sub-system" for that matter) to use as an Add-on, without also getting other necessities?? Additional power? Additional cables?? What? [I have two 2600s, each with only 2-drive floppy systems in a cabinet.] Your advice on this would be most appreciated.

Ed King, P.O. Box 787, Ithaca NY 14850

Ed: The Tandon drives come initially as a "bare" drive. No power supply, no cabinet. They are just what you had in your hand when you removed your drive for motor replacement. These can be used as direct replacements in Vector Graphic 2600s, System-3s, 3005s or 5005s - any Vector using a Tandon drive. The drive indeed gets its power from the computer. No cable changes or modifications are required for this replacement.

What DAMAN calls a "sub-system" is a "bare" drive put into an enclosure that contains its own power supply. When discussing a Vector Graphic application, it is possible to get power from the computer instead of using a new supply, but I've not tried that. These "sub-systems" are normally used as add-on drives. That is, to add a "C", or a "C" and "D" drive to your current "A" and "B". (Actually, it probably would be adding a "D" and "E" drive, with a reference to "C" being the access of the "B" drive as single-sided. It all depends on how you configure your system.) I use a single add-on with both my 3005 and 5005 (both have a hard disk and a single floppy) so I can do disk copies. I also feel safer when I write my hard-disk files to two floppies. I've written floppy files that I can't read back, even though the write process showed no errors.

Add-ons require a new data cable. Data cables are 34-wire flat cables with 34-pin female edge connectors attached at proper positions to attach to the present circuit board of the drives and the controller board. Obviously, if you place one or more drives outside the 2600's chassis, you'll need a longer cable.

Data cables seem to function well with lengths of up to 10 feet, but the shorter the better, I suppose. You can get "twisted" 34-wire flat cable, and can get shielded versions. I've had no trouble with plain flat cable, though. The cable parts can be purchased from most electronics stores, and certainly from Priority One Electronics. DAMAN can also make these cables for you. The current prices for Tandon 104M drives (100 TPI, 77 track, double-sided) are: Bare Drive...$247 One Drive in Encl...$335 Two Drives in Encl...$607 Shipping must be added to the above prices.

.....

CLASSIFIED

FOR SALE: Vector Graphic MZ with terminal and Centronics printer, and all manuals. $1000. Call Kirt Decker, (602) 247-4471

WANTED: Vector Graphic 3005 or Vector with 32Mb hard disk. Need two or three additional computers. Dr. Roger Parenteau, P.O. Box 2440, Vernon Ct 06066.

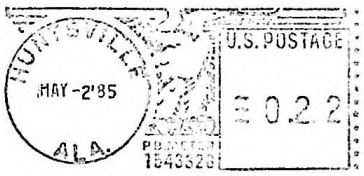
Roger - You might try calling Tom McGuire at The Used Computer Exchange, 4540 Mac Arthur Blvd. N.W., Suite 306, Washington D.C. 20007; (202) 337-1303.

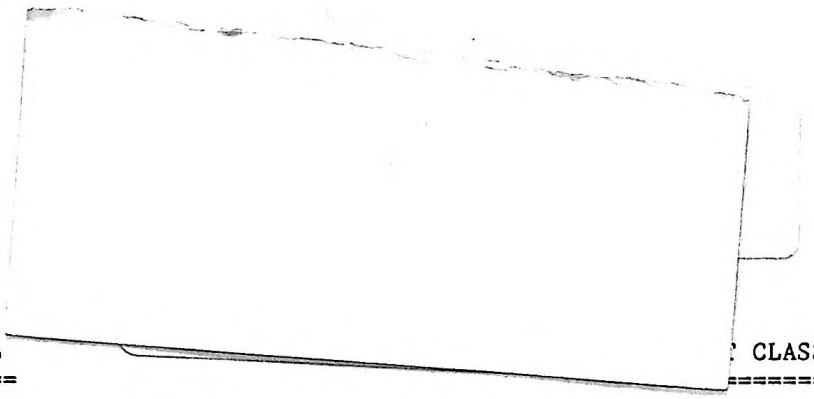
.....

****
* MICROPOLIS/VECTOR GRAPHIC USERS GROUP NEWSLETTER *
* Published Monthly by DAMAN, 604 Springwood Circle, Huntsville AL 35803 *
* Subscription rates: North America; $18/year Elsewhere; $25/year, Airmailed *
* *
* For Assistance with general information, MUG/Vector Graphic, DAMAN hardware *
* & software, Micropolis Basic, Basic/z, Micropolis drives & Micropolis parts-* *
* *
* Call Lynn or Buzz at (205)881-1697 during the Central Times of 9 AM to 9 PM.* *
* *
****

==========

MICROPOLIS/VECTOR GRAPHIC
USERS GROUP
A Division of DAMAN
604 Springwood Circle
Huntsville AL 35803
(205) 881-1697





FIRST CLASS MAIL ... CLASS MAIL

==========